Power exploration for embedded VLIW architectures

Full text Pdf (280 KB)

Source **International Conference on Computer Aided Design** archive
Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design table of contents
San Jose, California
SESSION: Session 10B: VLIW exploration and deisgn synthesis table of contents
Pages: 498 - 503
Year of Publication: 2000
ISBN:0-7803-6448-1

11/2000

Authors
Mariagiovanna Sami Politecnico di Milano, 20133 Milano, ITALY
Donatella Sciuto Politecnico di Milano, 20133 Milano, ITALY
Cristina Silvano Politecnico di Milano, 20133 Milano, ITALY
Vittorio Zaccaria Politecnico di Milano, 20133 Milano, ITALY

Sponsors : The IEEE Computer Society DATC
: IEEE Circuits & Systems Society
SIGDA: ACM Special Interest Group on Design Automation

Publisher IEEE Press Piscataway, NJ, USA

↑ ABSTRACT

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 A. Chandrakasan and R. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," Proc. of IEEE, 83(4), pp. 498-523, 1995.

2 M. Sami , D. Sciuto , C. Silvano , V. Zaccaria, Instruction-level power estimation for embedded VLIW cores, Proceedings of the eighth international workshop on Hardware/software codesign, p.34-38, May 2000, San Diego, California, United States

3 Vivek Tiwari , Sharad Malik , Andrew Wolfe, Power analysis of embedded software: a first step towards software power minimization, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.2 n.4, p.437-445, Dec. 1994

4 Mike Tien-Chien Lee , Masahiro Fujita , Vivek Tiwari , Sharad Malik, Power analysis and minimization techniques for embedded DSP software, IEEE Transactions on Very Large Scale

Refine Search

Search Results -

Terms	Documents
L10 AND compiler	31

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L11

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Monday, February 21, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=NO; OP=OR			
L11	L10 AND compiler	31	L11
L10	L9 and classification	40	L10
L9	L8 AND slot	462	L9
L8	l4 OR l3 OR l2 OR l1	2071	L8
L7	L5 and slot	1	L7
L6	L5 and classification	0	L6
L5	L4 and l3 and l2 and l1	3	L5
L4	instruction ADJ level ADJ parallelism	438	L4
L3	basic adj instruction	469	L3
L2	Very ADJ Long ADJ instruction ADJ Word	929	L2
L1	VLIW	1393	L1

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 3 of 3 returned.

☑ 1. Document ID: US 6760906 B1

L5: Entry 1 of 3 File: USPT Jul 6, 2004

US-PAT-NO: 6760906
DOCUMENT-IDENTIFIER: US 6760906 B1
** See image for Certificate of Correction **

TITLE: METHOD AND SYSTEM FOR PROCESSING PROGRAM FOR PARALLEL PROCESSING PURPOSES, STORAGE MEDIUM HAVING STORED THEREON PROGRAM GETTING PROGRAM PROCESSING EXECUTED FOR PARALLEL PROCESSING PURPOSES, AND STORAGE MEDIUM HAVING STORED THEREON INSTRUCTION SET TO BE EXECUTED IN PARALLEL

DATE-ISSUED: July 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Odani; Kensuke	Osaka			JP
Heishi; Taketo	Osaka			JP

US-CL-CURRENT: 717/149; 717/146, 717/152, 717/159

ABSTRACT:

A parallel data processing system is provided for increasing the program execution rate of a target machine. A parallelizer converts intermediate code, which has been generated by a compiler front end, into a parallelly executable form. An execution order determiner determines the order of the basic blocks to be executed. An expanded basic block parallelizer subdivides the intermediate code of the basic blocks into execution units, each of which is made up of parallelly executable instructions, following the order determined and on the basic block basis. When a particular one of the basic blocks is subdivided into execution units, an instruction belonging to the first execution unit of the next basic block, which has already been subdivided into execution units, is also used.

28 Claims, 27 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 12

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw De

☑ 2. Document ID: US 5680637 A

L5: Entry 2 of 3 File: USPT Oct 21, 1997

US-PAT-NO: 5680637
DOCUMENT-IDENTIFIER: US 5680637 A

TITLE: Computer having a parallel operating capability

DATE-ISSUED: October 21, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hotta; Takashi	Hitachi			JP
Nakatsuka; Yasuhiro	Hitachi			JP
Tanaka; Shigeya	Hitachi			JP
Yamada; Hiromichi	Hitachi			JP
Maejima; Hideo	Hitachi			JP

US-CL-CURRENT: 712/24; 712/206, 712/208, 712/215, 712/216, 712/23

ABSTRACT:

A RISC processor is arranged to reduce a code size, make the hardware less complicated, execute a plurality of operations for one machine cycle, and enhance the performance. The processor is capable of executing N instruction each having a short word length for indicating a single operation or an instruction having a long word length for indicating M (N<M) operations. When the number of operations to be executed in parallel is large, the long-word instruction is used. When it is small, the short-word instruction is used. A competition between the long-word instructions is detected by hardware and a competition between the short-word instructions only is detected by software. The simplification of the hardware brings about improvement of a machine cycle, improvement of a code cache hit ratio caused by the reduction of a code size and increase of the number of operations to be executed in parallel for the purpose of enhancing the performance.

7 Claims, 41 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 36

☐ 3. Document ID: US 5649135 A

L5: Entry 3 of 3 File: USPT Jul 15, 1997

US-PAT-NO: 5649135
DOCUMENT-IDENTIFIER: US 5649135 A

TITLE: Parallel processing system and method using surrogate instructions

DATE-ISSUED: July 15, 1997

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Pechanek; Gerald G.	Cary	NC		

Glossner; Clair John	Durham	NC	
Larsen; Larry D.	Raleigh	NC	
Vassiliadis; Stamatis	Zoetermeer		NL

US-CL-CURRENT: 712/200

ABSTRACT:

A parallel processing system and method is disclosed, which provides an improved instruction distribution mechanism for a parallel processing array. The invention broadcasts a basic instruction to each of a plurality of processor elements. Each processor element decodes the same instruction by combining it with a unique offset value stored in each respective processor element, to produce a derived instruction that is unique to the processor element. A first type of basic instruction results in the processor element performing a logical or control operation. A second type of basic instruction results in the generation of a pointer address. The pointer address has a unique address value because it results from combining the basic instruction with the unique offset value stored at the processor element. The pointer address is used to access an alternative instruction from an alternative instruction storage, for execution in the processor element. The alternative instruction is a very long instruction word, whose length is, for example, an integral multiple of the length of the basic instruction and contains much more information than can be represented by the basic instruction. A very long instruction word such as this is useful for providing parallel control of a plurality of primitive execution units that reside within the processor element. In this manner, a high degree of flexibility and versatility is attained in the operation of processor elements of a parallel processing array.

23 Claims, 28 Drawing figures
Exemplary Claim Number: 21
Number of Drawing Sheets: 26

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 31 of 31 returned.

☐ 1. Document ID: US 6789181 B1

L11: Entry 1 of 31 File: USPT Sep 7, 2004

US-PAT-NO: 6789181
DOCUMENT-IDENTIFIER: US 6789181 B1

TITLE: Safety net paradigm for managing two computer execution modes

DATE-ISSUED: September 7, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yates; John S.	Needham	MA		
Reese; David L.	Westborough	MA		
Van Dyke; Korbin S.	Sunol	CA		
Hohensee; Paul H.	Nashua	NH		

US-CL-CURRENT: 712/4; 714/37

ABSTRACT:

A method and computer for executing the method. A source program is translated into an object program, in a manner in which the translated object program has a different execution behavior than the source program. The translated object program is executed under a monitor capable of detecting any deviation from fully-correct interpretation before any side-effect of the different execution behavior is irreversibly committed. When the monitor detects the deviation, or when an interrupt occurs during execution of the object program, a state of the program is established corresponding to a state that would have occurred during an execution of the source program, and from which execution can continue. Execution of the source program continues primarily in a hardware emulator designed to execute instructions of an instruction set non-native to the computer.

48 Claims, 50 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 41

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 2. Document ID: US 6779107 B1

L11: Entry 2 of 31 File: USPT Aug 17, 2004

US-PAT-NO: 6779107
DOCUMENT-IDENTIFIER: US 6779107 B1

TITLE: Computer execution by opportunistic adaptation .

DATE-ISSUED: August 17, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yates; John S.	Needham	MA		

US-CL-CURRENT: 712/229; 712/41, 712/43

ABSTRACT:

A microprocessor chip and methods for execution by the microprocessor chip. Instruction pipeline circuitry has first and second correct modes for processing at least some instructions. A plurality of flags each correspond to a class of instruction occurring in the instruction pipeline circuitry. Pipeline control circuitry cooperates with the instruction pipeline circuitry, as part of the basic execution cycle of the computer, to maintain the value of the flags to record failures of an attempt to execute in the first mode two mode instructions of the corresponding respective instruction classes, to be triggered by a timer expiry to switch the value of the flags, thereby to switch the instruction pipeline circuitry from one of the processing modes to the other for the corresponding instruction class. The mode switch persists for instructions consecutively executed on behalf of a program that was in execution immediately before the timer expiry, beyond any exception handlers invoked consequent to the timer expiry. As each classified instruction comes up for execution in the instruction pipeline circuitry, the instruction pipeline circuitry executes the instruction in a mode determined, at least in part, by the value of the corresponding flag.

30 Claims, 50 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 41

☐ 3. Document ID: US 6615340 B1

L11: Entry 3 of 31 File: USPT Sep 2, 2003

US-PAT-NO: 6615340
DOCUMENT-IDENTIFIER: US 6615340 B1

TITLE: Extended operand management indicator structure and method

DATE-ISSUED: September 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wilmot, II; Richard Byron	Lafayette	CA	94549	

US-CL-CURRENT: 712/209; 712/217, 712/218

ABSTRACT:

Extended operand management indicators stored during initial program execution enable management and regulation of operand values and streamline their handling. Operand values are stored in new types of stores. Operand location management indicators indicate current operand value locations among various store types for selected operands. Indicated operand-forwarding policies for selected operands streamline forwarding of operand values from source instructions to value receiving target instructions. Indicated loop iterations of operand source instructions enable forwarding of operands over more than one loop iteration. Stride indicators indicate strides of program loop accesses to matrix operands. Inter-loop indicators enable forwarding of operand values from source loop instructions directly to target loop instructions. Constant or nearly constant operands are indicated to enable their storage in special caches. Operands used for cross-CPU serialization are indicated for special handling and storage in spin lock cache. Indicators of farthest back and farthest forward branches since operand last update are used to enhance the prediction of longer-range branch directions. Virtual predicate operand indicators streamline execution of densely branching program code. Stack operand indicators enable nullification of paired stack pointer increment-decrement operations to avoid serious operand serialization bottlenecks in very high issue rate machines.

15 Claims, 19 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 19

☐ 4. Document ID: US 6553513 B1

L11: Entry 4 of 31 File: USPT Apr 22, 2003

US-PAT-NO: 6553513
DOCUMENT-IDENTIFIER: US 6553513 B1

TITLE: Emulation suspend mode with differing response to differing classes of interrupts

DATE-ISSUED: April 22, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Swoboda; Gary L.	Sugarland	TX		
Matt; David R.	Missouri City	TX		

US-CL-CURRENT: 714/28; 714/25, 714/27, 714/30

ABSTRACT:

Emulation and debug circuitry is provided that can be incorporated into a variety of digital systems. A stop mode of operation is provided in which an associated processor stops processing instructions in response to a debug event. A real-time

mode of operation is provided in which the processor stops processing background instructions in response to a debug event, but in which high priority interrupts are still processed. Interrupts are classified and processed accordingly when the processor is stopped by a debug event. While suspended for a debug event, a frame counter keeps track of interrupt debug state if multiple interrupts occur. While running or suspended, the emulation circuitry can jam an instruction into the instruction register of the processor to cause processor resources to be read or written on behalf of the emulation circuitry. Read/write transactions are qualified by an expected frame count to maintain correspondence between test host software and multiple debug/interrupt events. An embodiment of a processor core is provided that is a programmable digital signal processor (DSP) with variable instruction length, offering both high code density and easy programming. Architecture and instruction set are optimized for low power consumption and high efficiency execution of DSP algorithms, such as for wireless telephones, as well as pure control tasks.

17 Claims, 34 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

☐ 5. Document ID: US 6549959 B1

L11: Entry 5 of 31 File: USPT Apr 15, 2003

US-PAT-NO: 6549959
DOCUMENT-IDENTIFIER: US 6549959 B1

TITLE: Detecting modification to computer memory by a DMA device

DATE-ISSUED: April 15, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yates; John S.	Needham	MA		
Reese; David L.	Westborough	MA		
Van Dyke; Korbin S.	Sunol	CA		

US-CL-CURRENT: 710/22; 710/1, 711/144, 711/146, 711/170

ABSTRACT:

A method and computer for executing the method. A CPU is programmed to execute first and second processes, the first process programmed to generate a second representation in a computer memory of information of the second process stored in the memory in a first representation. A main memory divided into pages for management by a virtual memory manager that uses a table stored in the memory. DMA (direct memory access) monitoring circuitry and/or software is designed to monitor DMA memory write transactions to a main memory of a computer by a DMA device of the computer; to detect when the first representation is overwritten by a DMA memory write transaction initiated by the second process, without the second process informing the first process of the DMA memory write transaction, the detecting guaranteed to occur no later than the next access of the second representation

following the DMA memory write transaction; to record an indication of a location in the main memory written by the DMA memory write transaction, the DMA monitoring circuitry designed to operate without being informed of the DMA memory write transaction by a CPU of the computer before initiation of the DMA memory write transaction, and to provide the indication to the CPU on request; and to report to the first process that the first representation is overwritten by a DMA memory write transaction. The DMA monitoring circuitry includes a plurality of registers outside the address space of the main memory, each register including an address tag and a vector of memory cells, and control circuitry designed to establish an association between a one of the plurality of registers with a region of the memory when a modification to the region is detected by setting the address tag of the one register to an approximation of the address of the region, and to set the values of the memory cells of the vector to record a fine indication of the address of a memory location modified, the control circuitry being operable without continuing supervisory control of a CPU of the computer. Circuitry is designed to record indications of modification to pages of the main memory into the registers. Read circuitry is designed to respond to a read request from the CPU by providing an address of a modified memory location. The virtual memory management tables do not provide backing store for the modification indications stored in the registers.

60 Claims, 50 Drawing figures
Exemplary Claim Number: 2
Number of Drawing Sheets: 41

☐ 6. Document ID: US 6549930 B1

L11: Entry 6 of 31 File: USPT Apr 15, 2003

US-PAT-NO: 6549930
DOCUMENT-IDENTIFIER: US 6549930 B1

TITLE: Method for scheduling threads in a multithreaded processor

DATE-ISSUED: April 15, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey A.	Menlo Park	CA		
Hicks, Jr.; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 718/104

ABSTRACT:

A method is provided for scheduling execution of a plurality of threads executed in a multithreaded processor. Resource utilizations of each of the plurality of threads are measured while the plurality of threads are concurrently executing in the multithreaded processor. Each of the plurality of threads is scheduled

according to the measured resource utilizations using a thread scheduler.

10 Claims, 21 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 20

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 7. Document ID: US 6493686 B1

L11: Entry 7 of 31 File: USPT Dec 10, 2002

US-PAT-NO: 6493686
DOCUMENT-IDENTIFIER: US 6493686 B1

TITLE: Computer implemented machine learning method and system including specifically defined introns

DATE-ISSUED: December 10, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Francone; Frank D.	Los Angeles	CA	90029	
Nordin; Peter	S-411 05 Goteborg			SE
Banzhaf; Wolfgang	Castrop-Rouxel			DE

US-CL-CURRENT: 706/12; 700/29, 700/48

ABSTRACT:

In a computer implemented learning and/or process control system, a computer model is constituted by the most currently fit entity in a population of computer program entities. The computer model defines fitness as a function of inputs and outputs. A computing unit accesses the model with a set of inputs, and determines a set of outputs for which the fitness is highest. This associates a sensory-motor (input-output) state with a fitness in a manner that might be termed "feeling".

The learning and/or control system preferably utilizes a Compiling Genetic Programming System (CGPS) in which one or more machine code entities such as functions are created which represent solutions to a problem and are directly executable by a computer. The programs are created and altered by a program in a higher level language such as "C" which is not directly executable, but requires translation into executable machine code through compilation, interpretation, translation, etc. The entities are initially created as an integer array that can be altered by the program as data, and are executed by the program by recasting a pointer to the array as a function type. The entities are evaluated by executing them with training data as inputs, and calculating fitnesses based on a predetermined criterion. The entities are then altered based on their fitnesses using a genetic machine learning algorithm by recasting the pointer to the array as a data (e.g. integer) type. This process is iteratively repeated until an end criterion is reached.

22 Claims, 55 Drawing figures
Exemplary Claim Number: 1

Number of Drawing Sheets: 49

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 8. Document ID: US 6397379 B1

L11: Entry 8 of 31 File: USPT May 28, 2002

US-PAT-NO: 6397379
DOCUMENT-IDENTIFIER: US 6397379 B1

TITLE: Recording in a program execution profile references to a memory-mapped active device

DATE-ISSUED: May 28, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yates, Jr.; John S.	Needham	MA		
Reese; David L.	Westborough	MA		
Van Dyke; Korbin S.	Sunol	CA		

US-CL-CURRENT: 717/140

ABSTRACT:

A method and a computer for execution of the method. As part of executing a stream of instructions, a series of memory loads is issued from a computer CPU to a bus, some directed to well-behaved memory and some directed to non-well-behaved devices in I/O space. Computer addresses are stored of instructions of the stream that issued memory loads to the non-well-behaved memory, the storage form of the recording allowing determination of whether the memory load was to well-behaved memory or not-well-behaved memory without resolution of any memory address stored in the recording.

47 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 41

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 9. Document ID: US 6298370 B1

L11: Entry 9 of 31 File: USPT Oct 2, 2001

US-PAT-NO: 6298370
DOCUMENT-IDENTIFIER: US 6298370 B1

TITLE: Computer operating process allocating tasks between first and second processors at run time based upon current processor load

DATE-ISSUED: October 2, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Tang; Jun	Richardson	TX		
So; John Ling Wing	Plano	TX		

US-CL-CURRENT: 718/102; 718/100

ABSTRACT:

A process of operating a computer system (100). The computer system (100) has a storage (HDD, 110) holding an operating system (OS) and an application program (APP.exe), a first processor (106) having an instruction set, and a second processor (1730) having a different instruction set. The process includes steps of 1) running (2424) at least some of the operating system (OS) on the first processor (106) so that the first processor (106) sets up for at least part of the application program at run time at least one second processor object (VSP OBJECT 1); and 2) concurrently running the second processor (3310) to access the second processor object (VSP OBJECT1) and thereby determine operations for the second processor (1730) to access second processor instructions for said part of the application program (APP.exe) and data to be processed according to said second processor instructions, and running (2436) the second processor (1730) to process the data according to said second processor instructions. Other processes, systems, devices and methods are also disclosed.

2 Claims, 159 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 104

☐ 10. Document ID: US 6195748 B1

L11: Entry 10 of 31 File: USPT Feb 27, 2001

US-PAT-NO: 6195748
DOCUMENT-IDENTIFIER: US 6195748 B1
**** See image for Certificate of Correction ****

TITLE: Apparatus for sampling instruction execution information in a processor pipeline

DATE-ISSUED: February 27, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

Leibholz; Daniel L.	Cambridge	MA
McLellan; Edward J.	Holliston	MA

US-CL-CURRENT: 712/227; 714/38, 714/45

ABSTRACT:

An apparatus is provided for sampling instructions in a processor pipeline of a computer system. The pipeline has a plurality of processing stages. Instructions are fetched into a first stage of the pipeline. A subset of the fetched instructions are identified as selected instructions. Event, latency, and state information of the system is sampled while any of the selected instructions are in any stage of the pipeline. Software is informed whenever any of the selected instructions leaves the pipeline to read the event and latency information.

49 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

☐ 11. Document ID: US 6179489 B1

L11: Entry 11 of 31 File: USPT Jan 30, 2001

US-PAT-NO: 6179489
DOCUMENT-IDENTIFIER: US 6179489 B1

TITLE: Devices, methods, systems and software products for coordination of computer main microprocessor and second microprocessor coupled thereto

DATE-ISSUED: January 30, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
So; John Ling Wing	Plano	TX		
Kerr; Jeffrey L.	Garland	TX		
Magee; Steven R.	Carrollton	TX		
Tang; Jun	Richardson	TX		

US-CL-CURRENT: 718/102

ABSTRACT:

A process is provided for operating a computer system (100) having a storage holding an operating system (OS) and an application program (APP.exe) and a third program (VSP Kernel), a first processor (106) having an instruction set, and a second processor (1730) having a different instruction set. The process includes a first step of running the first processor (106) to determine whether a part of the application shall be run on the first processor or the second processor and then establishing a second processor object (VSP OBJECT1) if said part shall be run on the second processor and the first processor (106) sending a message that the

second processor (1730) is to run said at least part of the application program. The third program establishes message handling functions and bus masters data transfer operations for the second processor between the host running the operating system and the second processor running the third program. The process concurrently runs the second processor to perform operations defined by the third program, including to access memory to detect the message that the second processor is to run said at least part of the application program, and runs the second processor (1730) to access the second processor object and thereby determine operations for the second processor to access second processor instructions for said part of the application program and data to be processed according to said second processor instructions.

3 Claims, 159 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 104

☐ 12. Document ID: US 6178492 B1

L11: Entry 12 of 31 File: USPT Jan 23, 2001

US-PAT-NO: 6178492
DOCUMENT-IDENTIFIER: US 6178492 B1
**** See image for Certificate of Correction ****

TITLE: Data processor capable of executing two instructions having operand interference at high speed in parallel

DATE-ISSUED: January 23, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Matsuo; Masahito	Tokyo			JP

US-CL-CURRENT: 712/23; 712/206, 712/212

ABSTRACT:

A data processor comprises an instruction decoding unit having two decoders decoding respective instructions of an instruction group consisting of a plurality of instructions including a first instruction and a second instruction succeeding the first instruction, and a judging unit judging whether or not a combination of the first instruction and the second instruction can be executed in parallel and a bus for transferring two data in parallel between an operand access unit and an integer operation unit. The data processor uses a superscalar technique. Two instructions having an operand interference can be executed in parallel at high speed and two instructions accessing a memory can be executed in parallel without considerable hardware.

30 Claims, 78 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 78

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 13. Document ID: US 6163840 A

L11: Entry 13 of 31 File: USPT Dec 19, 2000

US-PAT-NO: 6163840
DOCUMENT-IDENTIFIER: US 6163840 A
** See image for Certificate of Correction **

TITLE: Method and apparatus for sampling multiple potentially concurrent instructions in a processor pipeline

DATE-ISSUED: December 19, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		
Leibholz; Daniel L.	Cambridge	MA		
McLellan; Edward J.	Holliston	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 712/227

ABSTRACT:

An apparatus is provided for sampling multiple concurretly executing instructions in a processor pipeline of a system. The pipeline has a plurality of processing stages. The apparatus identifies multiple selected when the instructions are fetched into a first stage of the pipeline. A subset of the the multiple selected instructions to execute concurrently in the pipeline. State information of the system is sampled while any of the multiple selected instructions are in any stage of the pipeline. Software is informed whenever all of the selected instructions leave the pipeline so that the software can read any of the state information.

39 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. D

☐ 14. Document ID: US 6148396 A

L11: Entry 14 of 31 File: USPT Nov 14, 2000

US-PAT-NO: 6148396
DOCUMENT-IDENTIFIER: US 6148396 A

**** See image for Certificate of Correction ****

TITLE: Apparatus for sampling path history in a processor pipeline

DATE-ISSUED: November 14, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey	Menlo Park	CA		
Eustace; Robert A.	Redwood City	CA		
Hicks; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 712/227; 712/216, 714/47

ABSTRACT:

An apparatus is provided for collecting state information associated with an execution path of recently processed instructions in a processor pipeline of a computer system. The apparatus identifies a class of instructions to be sampled. Path-identifying state information of a currently processed instruction is sampled when the currently processed instruction belongs to the identified class of instructions. A shift register stores a predetermined number of entries storing selected state information, the shift register is simultaneously sampled along with additional state information about the instruction being executed at the time of sampling.

43 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

☐ 15. Document ID: US 6128607 A

L11: Entry 15 of 31 File: USPT Oct 3, 2000

US-PAT-NO: 6128607
DOCUMENT-IDENTIFIER: US 6128607 A

TITLE: Computer implemented machine learning method and system

DATE-ISSUED: October 3, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nordin; Peter	S-411 05 Goteborg			SE
Banzhaf; Wolfgang	44575 Castrop-Rauxel			DE

US-CL-CURRENT: 706/13

ABSTRACT:

One or more machine code entities such as functions are created which represent solutions to a problem and are directly executable by a computer. The programs are created and altered by a program in a higher level language such as "C" which is not directly executable, but requires translation into executable machine code through compilation, interpretation, translation, etc. The entities are initially created as an integer array that can be altered by the program as data, and are executed by the program by recasting a pointer to the array as a function type. The entities are evaluated by executing them with training data as inputs, and calculating fitnesses based on a predetermined criterion. The entities are then altered based on their fitnesses using a machine learning algorithm by recasting the pointer to the array as a data (e.g. integer) type. This process is iteratively repeated until an end criterion is reached. The entities evolve in such a manner as to improve their fitness, and one entity is ultimately produced which represents an optimal solution to the problem. Each entity includes a plurality of directly executable machine code instructions, a header, a footer, and a return instruction. The instructions include branch instructions which enable subroutines, leaf functions, external function calls, recursion, and loops. The system can be implemented on an integrated circuit chip, with the entities stored in high speed memory in a central processing unit.

202 Claims, 49 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 44

☐ 16. Document ID: US 6119075 A

L11: Entry 16 of 31 File: USPT Sep 12, 2000

US-PAT-NO: 6119075
DOCUMENT-IDENTIFIER: US 6119075 A

TITLE: Method for estimating statistics of properties of interactions processed by a processor pipeline

DATE-ISSUED: September 12, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		
Root; Stephen C.	Westborough	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 702/186; 712/237, 712/240

ABSTRACT:

Provided is a method for estimating statistics of properties of interactions among instructions processed in a pipeline of a computer system, the pipeline having a plurality of processing stages. Instructions are fetched into a first stage of the pipeline. A set of instructions are randomly selected from the fetched instructions, a subset of the set of selected instructions concurrently executing with each other. A distances between the set of selected instructions is specified, and state information of the computer system is recorded while the set of selected instructions is being processed by the pipeline. The recorded state information is communicated to software where it is statistically analyzed for a plurality of sets of selected instructions to estimate statistics of the interactions among sets of selected instructions.

34 Claims, 23 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 22

☐ 17. Document ID: US 6112289 A

L11: Entry 17 of 31 File: USPT Aug 29, 2000

US-PAT-NO: 6112289
DOCUMENT-IDENTIFIER: US 6112289 A
**** See image for Certificate of Correction ****

TITLE: Data processor

DATE-ISSUED: August 29, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Matsuo; Masahito	Tokyo			JP

US-CL-CURRENT: 712/23; 710/5

ABSTRACT:

A data processor comprises an instruction decoding unit having two decoders decoding respective instructions of an instruction group consisting of a plurality of instructions including a first instruction and a second instruction succeeding the first instruction, and a judging unit judging whether or not a combination of the first instruction and the second instruction can be executed in parallel and a bus for transferring two data in parallel between an operand access unit and an integer operation unit. The data processor uses a superscalar technique. Two instructions having an operand interference can be executed in parallel at high speed and two instructions accessing a memory can be executed in parallel without considerable hardware.

8 Claims, 78 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 78

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D.

☐ 18. Document ID: US 6105119 A

L11: Entry 18 of 31 File: USPT Aug 15, 2000

US-PAT-NO: 6105119
DOCUMENT-IDENTIFIER: US 6105119 A

TITLE: Data transfer circuitry, DSP wrapper circuitry and improved processor devices, methods and systems

DATE-ISSUED: August 15, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Kerr; Jeffrey L.	Garland	TX		
So; John Ling Wing	Plano	TX		
Magee; Steven R.	Carrollton	TX		

US-CL-CURRENT: 711/219; 710/110

ABSTRACT:

An integrated circuit (1720) includes a dual-port memory (3330.1) having a first memory port (Port A) and a second memory port (Port B), a bus interface block (5010) including bus master (5016) and bus slave circuitry (5018), and a byte-channeling block (5310) coupled between the first memory port (Port A) and the bus interface block (5010) operable to convert non-aligned data addresses into aligned data. Advantageously, this invention includes a single bus master serving all application hardware. This relieves the host of the extra burden of communicating to slave circuits, reducing host I/O MIPS significantly. The digital signal processor with an ASIC wrapper of this invention together provide super-bus-mastering to access the entire memory space in the system, including the entire virtual memory space accessible by the host processor. Other processes, systems, devices and methods are also disclosed.

2 Claims, 159 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 104

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D.

☐ 19. Document ID: US 6092180 A

L11: Entry 19 of 31 File: USPT Jul 18, 2000

US-PAT-NO: 6092180
DOCUMENT-IDENTIFIER: US 6092180 A

TITLE: Method for measuring latencies by randomly selected sampling of the instructions while the instruction are executed

DATE-ISSUED: July 18, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Anderson; Jennifer-Ann M.	Palo Alto	CA		
Dean; Jeffrey	Menlo Park	CA		
Hicks, Jr.; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 712/200; 712/202, 712/216, 717/158, 717/159

ABSTRACT:

In a method for scheduling instructions executed in a computer system including a processor and a memory subsystem, pipeline latencies and resource utilization are measured by sampling hardware while the instructions are executing. The instructions are then scheduled according to the measured latencies and resource utilizations using an instruction scheduler.

18 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 20

☐ 20. Document ID: US 6070009 A

L11: Entry 20 of 31 File: USPT May 30, 2000

US-PAT-NO: 6070009
DOCUMENT-IDENTIFIER: US 6070009 A

TITLE: Method for estimating execution rates of program execution paths

DATE-ISSUED: May 30, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Dean; Jeffrey	Menlo Park	CA		
Eustace; Robert A.	Redwood City	CA		
Hicks; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 717/130; 717/132, 717/154, 717/159

ABSTRACT:

A method is provided for estimating execution rates of program executions paths. The method samples path-identifying state information of selected instructions while executing the program in a processor. A control flow graph of the program is supplied, the control flow graph includes a plurality of path segments. The control flow graph is analyzed using the path-identifying state information to identify a set of path segments that are consistent with the sampled state information. The set of paths segments can be counted to determine their relative execution frequencies.

22 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 20

☐ 21. Document ID: US 6000044 A

L11: Entry 21 of 31 File: USPT Dec 7, 1999

US-PAT-NO: 6000044
DOCUMENT-IDENTIFIER: US 6000044 A

TITLE: Apparatus for randomly sampling instructions in a processor pipeline

DATE-ISSUED: December 7, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		
Leibholz; Daniel L.	Cambridge	MA		
McLellan; Edward J.	Holliston	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 714/47; 702/182, 702/183, 702/186, 712/233, 712/244, 713/502, 714/37, 714/39, 714/45

ABSTRACT:

An apparatus is provided for sampling instructions in a processor pipeline of a system. The pipeline has a plurality of processing stages. The apparatus includes a fetch unit for fetching instructions into a first stage of the pipeline. Certain randomly selected instructions are identified, and state information of the system is sampled while a particular selected instruction is in any stage of the pipeline. Software is informed when the particular selected instruction leaves the pipeline so that the software can read any of the sampled state information.

29 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 22. Document ID: US 5964867 A

L11: Entry 22 of 31 File: USPT Oct 12, 1999

US-PAT-NO: 5964867
DOCUMENT-IDENTIFIER: US 5964867 A
** See image for Certificate of Correction **

TITLE: Method for inserting memory prefetch operations based on measured latencies in a program optimizer

DATE-ISSUED: October 12, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Anderson; Jennifer-Ann M.	Palo Alto	CA		
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		
Waldspurger; Carl A.	Atherton	CA		
Weihl; William E.	San Francisco	CA		

US-CL-CURRENT: 712/219; 712/207, 712/216

ABSTRACT:

A method is provided for optimizing a program by inserting memory prefetch operations in the program executing in a computer system. The computer system includes a processor and a memory. Latencies of instructions of the program are measured by hardware while the instructions are processed by a pipeline of the processor. Memory prefetch instructions are automatically inserted in the program based on the measured latencies to optimize execution of the program. The latencies measure the time from when a load instructions issues a request for data to the memory until the data are available in the processor. A program optimizer uses the measured latencies to estimate the number of cycles that elapse before data of a memory operation are available.

7 Claims, 21 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw D

☐ 23. Document ID: US 5946674 A

L11: Entry 23 of 31 File: USPT Aug 31, 1999

US-PAT-NO: 5946674

DOCUMENT-IDENTIFIER: US 5946674 A

TITLE: Turing complete computer implemented machine learning method and system

DATE-ISSUED: August 31, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Nordin; Peter	Goteborg			SE
Banzhaf; Wolfgang	Castrop-Rauxel			DE

US-CL-CURRENT: 706/13

ABSTRACT:

One or more machine code entities such as functions are created which represent solutions to a problem and are directly executable by a computer. The programs are created and altered by a program in a higher level language such as "C" which is not directly executable, but requires translation into executable machine code through compilation, interpretation, translation, etc. The entities are initially created as an integer array that can be altered by the program as data, and are executed by the program by recasting a pointer to the array as a function type. The entities are evaluated by executing them with training data as inputs, and calculating fitnesses based on a predetermined criterion. The entities are then altered based on their fitnesses using a machine learning algorithm by recasting the pointer to the array as a data (e.g. integer) type. This process is iteratively repeated until an end criterion is reached. The entities evolve in such a manner as to improve their fitness, and one entity is ultimately produced which represents an optimal solution to the problem. Each entity includes a plurality of directly executable machine code instructions, a header, a footer, and a return instruction. The instructions include branch instructions which enable subroutines, leaf functions, external function calls, recursion, and loops. The system can be implemented on an integrated circuit chip, with the entities stored in high speed memory in a central processing unit.

18 Claims, 49 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 44

☐ 24. Document ID: US 5946673 A

L11: Entry 24 of 31 File: USPT Aug 31, 1999

US-PAT-NO: 5946673
DOCUMENT-IDENTIFIER: US 5946673 A

TITLE: Computer implemented machine learning and control system

DATE-ISSUED: August 31, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Francone; Frank D.	Los Angeles	CA	90029	
Nordin; Peter	S-411 05 Gothenburg			SE
Banzhaf; Wolfgang	Castrop-Rauxel			DE

US-CL-CURRENT: 706/13; 700/246, 701/301

ABSTRACT:

In a computer implemented learning and/or process control system, a computer model is constituted by the most currently fit entity in a population of computer program entities. The computer model defines fitness as a function of inputs and outputs. A computing unit accesses the model with a set of inputs, and determines a set of outputs for which the fitness is highest. This associates a sensory-motor (input-output) state with a fitness in a manner that might be termed "feeling". The learning and/or control system preferably utilizes a compiling Genetic Programming system (CGPS) in which one or more machine code entities such as functions are created which represent solutions to a problem and are directly executable by a computer. The programs are created and altered by a program in a higher level language such as "C" which is not directly executable, but requires translation into executable machine code through compilation, interpretation, translation, etc. The entities are initially created as an integer array that can be altered by the program as data, and are executed by the program by recasting a pointer to the array as a function type. The entities are evaluated by executing them with training data as inputs, and calculating fitnesses based on a predetermined criterion. The entities are then altered based on their fitnesses using a genetic machine learning algorithm by recasting the pointer to the array as a data (e.g. integer) type. This process is iteratively repeated until an end criterion is reached.

68 Claims, 55 Drawing figures
Exemplary Claim Number: 25
Number of Drawing Sheets: 49

☐ 25. Document ID: US 5923872 A

L11: Entry 25 of 31 File: USPT Jul 13, 1999

US-PAT-NO: 5923872
DOCUMENT-IDENTIFIER: US 5923872 A

TITLE: Apparatus for sampling instruction operand or result values in a processor pipeline

DATE-ISSUED: July 13, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Chrysos; George Z.	Marlborough	MA		
Dean; Jeffrey	Menlo Park	CA		
Hicks; James E.	Newton	MA		

Waldspurger; Carl A.	Atherton	CA
Weihl; William E.	San Francisco	CA

US-CL-CURRENT: 712/244; 712/23

ABSTRACT:

An apparatus is provided for sampling values of operands of instructions in a processor pipeline of a system, the pipeline having a plurality of processing stages. Instructions are fetched into a first stage of the pipeline. Any one of the fetched instructions are identified as a particular selected instruction. Values of results computed during the processing of the particular selected instruction are recorded in a sampling record along with state information identifying the particular selected instruction. Software is informed whenever the particular selected instruction leaves the pipeline to read the recorded values and state information.

32 Claims, 22 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 21

☐ 26. Document ID: US 5909559 A

L11: Entry 26 of 31 File: USPT Jun 1, 1999

US-PAT-NO: 5909559
DOCUMENT-IDENTIFIER: US 5909559 A

TITLE: Bus bridge device including data bus of first width for a first processor, memory controller, arbiter circuit and second processor having a different second data width

DATE-ISSUED: June 1, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
So; John Ling Wing	Plano	TX		

US-CL-CURRENT: 710/307

ABSTRACT:

An integrated circuit (2210) provides on a single chip for use with a first processor (106) off-chip, the following combination: first terminals (of 2232) for first processor-related signals and defining a first data width (32-bit), second terminals for external bus-related signals (PCI), third terminals for memory-related signals (of 2258), and a DRAM memory controller (2250) connected to the third terminals. Further on chip is provided an arbiter circuit (2230), a bus bridge circuit (2236) coupled to the DRAM memory controller and to the second terminals, the bus bridge (2236) also coupled to the arbiter (2230), a second processor (2224) having a second data width (16-bit), and a bus interface circuit

(2220) coupling the second data width of the second processor (2224) to the first data width. The bus interface circuit (2220) further has bus master and bus slave circuitry coupled between the second processor (2224) and the arbiter circuit (2230). The bus bridge (2236), the bus interface (2220) and the first terminals and the DRAM memory controller (2250) have datapaths selectively interconnected in response to the arbiter circuit (2230). Other devices, systems and methods are also disclosed.

5 Claims, 159 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 104

☐ 27. Document ID: US 5884060 A

L11: Entry 27 of 31 File: USPT Mar 16, 1999

US-PAT-NO: 5884060
DOCUMENT-IDENTIFIER: US 5884060 A

TITLE: Processor which performs dynamic instruction scheduling at time of execution within a single clock cycle

DATE-ISSUED: March 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Vegesna; Anantakotiraju	Austin	TX		
Avula; Jayachandra B.	Austin	TX		
Jewett; Peter H.	Austin	TX		
Mundkur; Yatin G.	Austin	TX		
Naik; Vinay J.	Austin	TX		
Monaco; James E.	Austin	TX		

US-CL-CURRENT: 712/215

ABSTRACT:

An apparatus and method for scheduling the execution of one or more of a sequence of instructions for superscalar execution by a central processing unit during a single clock cycle of the processor clock is disclosed wherein the scheduling process is performed in a manner which does not dictate the duration of the processor clock period. During the decode stage of the processor pipeline, the instructions are classified, decoded, and data and resource dependencies are detected and resolved for operand access, with these processes being performed virtually in parallel so that the instructions can be appropriately scheduled for execution at the beginning of the next processor clock cycle. Because of the parallel nature of the scheduling process, scheduling can be performed and completed fast enough that processes other than instruction scheduling will dictate the minimum processor clock period.

15 Claims, 24 Drawing figures

Exemplary Claim Number: 1
Number of Drawing Sheets: 23

Day : Monday
Date: 2/21/2005

Time: 22:41:33

PALM INTRANET

Inventor Information for 10/053707

Inventor Name	City	State/Country
KAMIGATA, TERUHIKO	KAWASAKI	JAPAN
MIYAKE, HIDEO	KAWASAKI	JAPAN

Appln Info | Contents | Petition Info | Atty/Agent Info | Continuity Data | Foreign Data

Search Another: Application# [ ] Search **or Patent#** [ ] Search

PCT / [ ] / [ ] Search **or PG PUBS #** [ ] Search

Attorney Docket # [ ] Search

Bar Code # [ ] Search

To go back use Back button on your browser toolbar.

Back to PALM | ASSIGNMENT | OASIS | Home page

Inventor name search
ACM

Patent Assignment Abstract of Title

Total Assignments: 1

Application #: 10053707 **Filing Dt:** 01/24/2002 **Patent #:** NONE **Issue Dt:**

PCT #: NONE **Publication #:** US20020161986 **Pub Dt:** 10/31/2002

Inventors: Teruhiko Kamigata, Hideo Miyake

Title: Instruction processing method for verifying basic instruction arrangement in VLIW instruction for variable length VLIW processor

Assignment: 1

Reel/Frame:	Received:	Recorded:	Mailed:	Pages:
012524/0526	02/04/2002	01/24/2002	03/27/2002	2

Conveyance: ASSIGNMENT OF ASSIGNORS INTEREST (SEE DOCUMENT FOR DETAILS).

Assignors: KAMIGATA, TERUHIKO **Exec Dt:** 01/11/2002

MIYAKE, HIDEO **Exec Dt:** 01/11/2002

Assignee: FUJITSU LIMITED
NAKAHARA-KU, KAWASAKI-SHI
1-1, KAMIKODANAKA 4-CHOME
KANAGAWA 211-8588, JAPAN

Correspondent: STAAS & HALSEY LLP
H.J. STAAS
700 ELEVENTH STREET, N.W., SUITE 500
WASHINGTON, D.C. 20001

Search Results as of: 2/21/2005 10:41:40 P.M.

Day : Monday
Date: 2/21/2005

PALM INTRANET

Time: 22:41:48

Inventor Name Search Result

Your Search was:

Last Name = KAMIGATA
First Name = TERUHIKO

Application#	Patent#	Status	Date Filed	Title	Inventor Name
09666853	Not Issued	061	09/20/2000	INFORMATION PROCESSING DEVICE	KAMIGATA, TERUHIKO
09671117	Not Issued	095	09/28/2000	METHOD OF CONTROLLING AND ADDRESSING A CACHE MEMORY WHICH ACTS AS A RANDOM ADDRESS MEMORY TO INCREASE AN ACCESS SPEED TO A MAIN MEMORY	KAMIGATA, TERUHIKO
10053707	Not Issued	030	01/24/2002	INSTRUCTION PROCESSING METHOD FOR VERIFYING BASIC INSTRUCTION ARRANGEMENT IN VLIW INSTRUCTION FOR VARIABLE LENGTH VLIW PROCESSOR	KAMIGATA, TERUHIKO
10603093	Not Issued	030	06/25/2003	METHOD OF AND APPARATUS FOR CREATING LOAD MODULE AND COMPUTER PRODUCT	KAMIGATA, TERUHIKO

Inventor Search Completed: No Records to Display.

Search Another: Inventor Last Name KAMIGATA First Name TERUHIKO Search

To go back use Back button on your browser toolbar.

Back to PALM | ASSIGNMENT | OASIS | Home page

Day : Monday
Date: 2/21/2005

PALM INTRANET

Time: 22:42:03

Inventor Name Search Result

Your Search was:

Last Name = MIYAKE
First Name = HIDEO

Application#	Patent#	Status	Date Filed	Title	Inventor Name
06045521	4255553	150	06/04/1979	POWDER COATING COMPOSITION	MIYAKE, HIDEO
06109736	4290938	150	01/04/1980	THERMOSETTING INJECTION MOLDING COMPOUND	MIYAKE, HIDEO
06217629	4340519	150	12/18/1980	POLYESTER RESIN AQUEOUS DISPERSION	MIYAKE, HIDEO
06219568	4379039	150	12/24/1980	ULTRAVIOLET CURABLE RESIN COMPOSITION	MIYAKE, HIDEO
06527934	4551215	150	08/30/1983	CURABLE RESIN COMPOSITION	MIYAKE, HIDEO
06642863	4586996	150	08/21/1984	SURFACE HARDNER FOR NYLON LENS	MIYAKE, HIDEO
06776173	4666756	150	09/05/1985	PRINTED TRANSFER PAPER FOR DECORATING POTTERY	MIYAKE, HIDEO
07103456	4888208	150	10/01/1987	CERAMIC SUBSTRATE FOR PRINTED CIRCUITS AND PRODUCTION THEREOF	MIYAKE, HIDEO
07674570	5118576	150	03/25/1991	MATERIAL FOR EXPANDED GRAPHITE GASKET	MIYAKE, HIDEO
07945388	5346975	150	09/16/1992	LIGHT-SENSITIVE COMPOSITION	MIYAKE, HIDEO
08087731	5429843	250	07/19/1993	VAPOR DEPOSITION FOR FORMATION OF PLATING LAYER	MIYAKE, HIDEO
08127733	Not Issued	161	09/28/1993	PHOTOSENSITIVE COMPOSITION	MIYAKE, HIDEO
08155373	5631090	150	11/22/1993	IRON-BASED MATERIAL HAVING EXCELLENT OXIDATION RESISTANCE AT ELEVATED TEMPERATURES AND PROCESS FOR THE PRODUCTION THEREOF	MIYAKE, HIDEO

08467637	5612090	150	06/06/1995	IRON-BASED MATERIAL HAVING EXCELLENT OXIDATION RESISTANCE AT ELEVATED TEMPERATURES AND PROCESS FOR THE PRODUCTION THEREOF	MIYAKE, HIDEO
09173719	6573022	150	10/16/1998	POSITIVE TYPE PHOTOSENSITIVE IMAGE-FORMING MATERIAL FOR AN INFRARED LASER AND A POSITIVE TYPE PHOTOSENSITIVE COMPOSITION FOR AN INFRARED LASER	MIYAKE, HIDEO
09421535	6340551	150	10/20/1999	POSITIVE TYPE PHOTOSENSITIVE IMAGE-FORMING MATERIAL FOR USE WITH AN INFRARED LASER	MIYAKE, HIDEO
09654527	Not Issued	061	09/01/2000	SELECTIVE INSTRUCTION ISSUING PARALLEL PROCESSOR	MIYAKE, HIDEO
09657349	6775762	150	09/07/2000	PROCESSOR AND PROCESSOR SYSTEM	MIYAKE, HIDEO
09668383	Not Issued	161	09/25/2000	DATA PROCESSING APPARATUS AND METHOD OF CONTROLLING THE SAME	MIYAKE, HIDEO
09671117	Not Issued	095	09/28/2000	METHOD OF CONTROLLING AND ADDRESSING A CACHE MEMORY WHICH ACTS AS A RANDOM ADDRESS MEMORY TO INCREASE AN ACCESS SPEED TO A MAIN MEMORY	MIYAKE, HIDEO
09678732	6681280	150	10/04/2000	INTERRUPT CONTROL APPARATUS AND METHOD SEPARATELY HOLDING RESPECTIVE OPERATION INFORMATION OF A PROCESSOR PRECEDING A NORMAL OR A BREAK INTERRUPT	MIYAKE, HIDEO
09736357	Not Issued	094	12/15/2000	PROCESSOR AND METHOD OF CONTROLLING THE SAME	MIYAKE, HIDEO
09741802	Not Issued	061	12/22/2000	INFORMATION PROCESSING UNIT, AND EXCEPTION	MIYAKE, HIDEO

				PROCESSING METHOD FOR SPECIFIC APPLICATION-PURPOSE OPERATION INSTRUCTION	
09768630	Not Issued	041	01/25/2001	COMPUTER WITH HIGH-SPEED CONTEXT SWITCHING	MIYAKE, HIDEO
09938496	Not Issued	080	08/27/2001	COMPUTER AND CONTROL METHOD OF THE COMPUTER	MIYAKE, HIDEO
09993634	Not Issued	163	11/27/2001	POSITIVE TYPE PHOTOSENSITIVE IMAGE-FORMING MATERIAL FOR AN INFRARED LASER AND A POSITIVE TYPE PHOTOSENSITIVE COMPOSITION FOR AN INFRARED LASER	MIYAKE, HIDEO
09996892	6841330	150	11/30/2001	PLANOGRAPHIC PRINTING PLATE PRECURSOR	MIYAKE, HIDEO
10053707	Not Issued	030	01/24/2002	INSTRUCTION PROCESSING METHOD FOR VERIFYING BASIC INSTRUCTION ARRANGEMENT IN VLIW INSTRUCTION FOR VARIABLE LENGTH VLIW PROCESSOR	MIYAKE, HIDEO
10082099	Not Issued	168	02/26/2002	POSITIVE TYPE PHOTOSENSITIVE IMAGE-FORMING MATERIAL FOR AN INFRARED LASER AND A POSITIVE TYPE PHOTOSENSITIVE COMPOSITION FOR AN INFRARED LASER	MIYAKE, HIDEO
10173820	Not Issued	071	06/19/2002	LITHOGRAPHIC PRINTING PLATE PRECURSOR AND PRODUCTION METHOD OF LITHOGRAPHIC PRINTING PLATE	MIYAKE, HIDEO
10179390	Not Issued	041	06/26/2002	PLANOGRAPHIC PRINTING PLATE PRECURSOR	MIYAKE, HIDEO
10190545	Not Issued	092	07/09/2002	LITHOGRAPHIC PRINTING PLATE PRECURSOR AND PRODUCTION METHOD OF LITHOGRAPHIC PRINTING PLATE	MIYAKE, HIDEO

10372242	Not Issued	071	02/25/2003	PROCESS FOR PRODUCING PRINTING PLATE PRECURSOR	MIYAKE, HIDEO
10603093	Not Issued	030	06/25/2003	METHOD OF AND APPARATUS FOR CREATING LOAD MODULE AND COMPUTER PRODUCT	MIYAKE, HIDEO
10692800	Not Issued	030	10/27/2003	INTERRUPT CONTROL APPARATUS AND METHOD	MIYAKE, HIDEO
10761099	Not Issued	019	01/21/2004	POSITIVE TYPE PHOTOSENSITIVE IMAGE-FORMING MATERIAL FOR USE WITH AN INFRARED LASER	MIYAKE, HIDEO
09354701	6419364	150	07/16/1999	PROJECTION DISPLAY DEVICE	MIYAKE, HIDEOKI
10180341	6702444	150	06/27/2002	PROJECTION DISPLAY DEVICE	MIYAKE, HIDEOKI

Inventor Search Completed: No Records to Display.

Search Another: Inventor **Last Name** MIYAKE **First Name** HIDEO Search

To go back use Back button on your browser toolbar.

Back to PALM | ASSIGNMENT | OASIS | Home page

Terms used **VLIW basic instruction classification slot** Found **44,049** of **150,885**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window
Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Optimal integrated code generation for clustered VLIW architectures
Christoph Kessler, Andrzej Bednarski
June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems**, Volume 37 Issue 7
Full text available: pdf(227.07 KB) Additional Information: full citation, abstract, references, index terms

In contrast to standard compilers, generating code for DSPs can afford spending considerable resources in time and space on optimizations. Generating efficient code for irregular architectures requires an integrated method that optimizes simultaneously for instruction selection, instruction scheduling, and register allocation.We describe a method for fully integrated optimal code generation based on dynamic programming. We introduce the concept of *residence classes* and *space profiles*
***Keywords**: dynamic programming, instruction scheduling, instruction selection, integrated code generation, register allocation, space profile*

2 Resource usage models for instruction scheduling: two new models and a classification
V. Janaki Ramanan, R. Govindarajan
May 1999 **Proceedings of the 13th international conference on Supercomputing**
Full text available: pdf(1.15 MB) Additional Information: full citation, references, index terms

3 Compiler optimization on VLIW instruction scheduling for low power
Chingren Lee, Jenq Kuen Lee, Tingting Hwang, Shi-Chun Tsai
April 2003 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 8 Issue 2
Full text available: pdf(175.72 KB) Additional Information: full citation, abstract, references, index terms, review

In this article, we investigate compiler transformation techniques regarding the problem of scheduling VLIW instructions aimed at reducing power consumption of VLIW architectures in the instruction bus. The problem can be categorized into two types: horizontal scheduling and vertical scheduling. For the case of horizontal scheduling, we propose a bipartite-matching scheme for instruction scheduling. We prove that our greedy bipartite-matching scheme always gives the optimal switching activities ...

Keywords: Compilers, VLIW instruction scheduling, instruction bus optimizations, low-power optimization

4 Instruction fetch mechanisms for VLIW architectures with compressed encodings

Thomas M. Conte, Sanjeev Banerjia, Sergei Y. Larin, Kishore N. Menezes, Sumedh W. Sathaye

December 1996 **Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.34 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

VLIW architectures use very wide instruction words in conjunction with high bandwidth to the instruction cache to achieve multiple instruction issue. This report uses the TINKER experimental testbed to examine instruction fetch and instruction cache mechanisms for VLIWs. A compressed instruction encoding for VLIWs is defined and a classification scheme for i-fetch hardware for such an encoding is introduced. Several interesting cache and i-fetch organizations are described and evaluated through ...

Keywords: TINKER experimental testbed, VLIW architectures, compressed encodings, compressed instruction encoding, i-fetch hardware, instruction cache, instruction fetch mechanisms, instruction words, multiple instruction issue, parallel architectures, silo cache, trace-driven simulations

5 Scheduling time-constrained instructions on pipelined processors

Allen Leung, Krishna V. Palem, Amir Pnueli

January 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 1

Full text available: pdf(357.07 KB) Additional Information: full citation, abstract, references, index terms

In this work we investigate the problem of scheduling instructions on idealized microprocessors with multiple pipelines, in the presence of precedence constraints, release-times, deadlines, and latency constraints. A latency of lij specifies that there must be at least lij time-steps between the completion time of instruction i and the start time of instruction j. A latency of

6 ILP-based Instruction Scheduling for IA-64

Daniel Kästner, Sebastian Winkel

August 2001 **ACM SIGPLAN Notices**, Volume 36 Issue 8

Full text available: pdf(369.18 KB) Additional Information: full citation, abstract, references, citings, index terms

The IA-64 architecture has been designed as a synthesis of VLIW and superscalar design principles. It incorporates typical functionality known from embedded processors as multiply/accumulate units and SIMD operations for 3D graphics operations. In this paper we present an ILP formulation for the problem of instruction scheduling for IA-64. In order to obtain a feasible schedule it is necessary to model the data dependences, resource constraints as well as additional encoding restrictions&mdas ...

7 Compiler scheduling: Effective instruction scheduling techniques for an interleaved cache clustered VLIW processor

Enric Gibert, Jesús Sánchez, Antonio González

November 2002 **Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.13 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Clustering is a common technique to overcome the wire delay problem incurred by the evolution of technology. Fully-distributed architectures, where the register file, the functional units and the data cache are partitioned, are particularly effective to deal with these constraints and besides they are very scalable. In this paper effective instruction scheduling techniques for a clustered VLIW processor with a word-interleaved cache are proposed. Such scheduling techniques rely on: (i) loop unro ...

8 Software pipelining: an effective scheduling technique for VLIW machines

M. Lam

June 1988 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1988 conference on Programming Language design and Implementation**, Volume 23 Issue 7

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper shows that software pipelining is an effective and viable scheduling technique for VLIW processors. In software pipelining, iterations of a loop in the source program are continuously initiated at constant intervals, before the preceding iterations complete. The advantage of software pipelining is that optimal performance can be achieved with compact object code. This paper extends previous results of software pipelining in two ways: First, this paper shows that by usi ...

9 Efficient DAG construction and heuristic calculation for instruction scheduling

Mark Smotherman, Sanjay Krishnamurthy, P. S. Aravind, David Hunnicutt

September 1991 **Proceedings of the 24th annual international symposium on Microarchitecture**

Full text available: pdf(992.45 KB) Additional Information: full citation, references, citings, index terms

10 Scheduling time-critical instructions on RISC machines

Krishna V. Palem, Barbara B. Simons

September 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 4

Full text available: pdf(1.89 MB) Additional Information: full citation, abstract, references, citings, index terms

We present a polynomial time algorithm for constructing a minimum completion time schedule of instructions from a basic block on RISC machines such as the Sun SPARC, the IBM 801, the Berkeley RISC machine, and the HP Precision Architecture. Our algorithm can be used as a heuristic for RISC processors with longer pipelines, for which there is no known optimal algorithm. Our algorithm can also handle time-critical instructions, which are instructions that have to be completed by a specific ti ...

Keywords: NP-complete, RISC machine scheduling, compiler optimization, deadline, greedy algorithm, instruction scheduling, latency, pipeline processor, register allocation

11 A survey of processors with explicit multithreading

Theo Ungerer, Borut Robič, Jurij Šilc

March 2003 **ACM Computing Surveys (CSUR)**, Volume 35 Issue 1

Full text available: pdf(920.16 KB) Additional Information: full citation, abstract, references, index terms

Hardware multithreading is becoming a generally applied technique in the next generation of microprocessors. Several multithreaded processors are announced by industry or already into production in the areas of high-performance microprocessors, media, and network processors.A multithreaded processor is able to pursue two or more threads of control in parallel within the processor pipeline. The contexts of two or more threads of control are

often stored in separate on-chip register sets. Unused i ...

Keywords: Blocked multithreading, interleaved multithreading, simultaneous multithreading

12 Code reuse in an optimizing compiler

Ali-Reza Adl-Tabatabai, Thomas Gross, Guei-Yuan Lueh

October 1996 **ACM SIGPLAN Notices , Proceedings of the 11th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 31 Issue 10

Full text available: pdf(1.97 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes how the cmcc compiler reuses code---both internally (reuse between different modules) and externally (reuse between versions for different target machines). The key to reuse are the application frameworks developed for global data-flow analysis, code generation, instruction scheduling, and register allocation.The code produced by cmcc is as good as the code produced by the native compilers for the MIPS and SPARC, although significantly less resources have been spent on cmcc ...

13 Measurement and evaluation of the MIPS architecture and processor

Thomas R. Gross, John L. Hennessy, Steven A. Przybylski, Christopher Rowen

August 1988 **ACM Transactions on Computer Systems (TOCS)**, Volume 6 Issue 3

Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, citings, index terms, review

MIPS is a 32-bit processor architecture that has been implemented as an nMOS VLSI chip. The instruction set architecture is RISC-based. Close coupling with compilers and efficient use of the instruction set by compiled programs were goals of the architecture. The MIPS architecture requires that the software implement some constraints in the design that are normally considered part of the hardware implementation. This paper presents experimental results on the effectiveness of this processor ...

14 Instruction path coprocessors

Yuan Chou, John Paul Shen

May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture**, Volume 28 Issue 2

Full text available: pdf(134.64 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents the concept of an Instruction Path Coprocessor (I-COP), which is a programmable on-chip coprocessor, with its own mini-instruction set, that operates on the core processor's instructions to transform them into an internal format that can be more efficiently executed. It is located off the critical path of the core processor to ensure that it does not negatively impact the core processor's cycle time or pipeline depth. An I-COP is highly versatile and can be used ...

15 Scheduling time-critical instructions on RISC machines

Krishna Palem, Barbara Simons

December 1989 **Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Full text available: pdf(1.17 MB) Additional Information: full citation, abstract, references, citings, index terms

An instruction or a set of instructions can be considered time critical if their execution is required to free up a resource. Time critical instructions might be used to make shared

resources such as registers more quickly available for reuse; or they might be used for real time computations, portions of which are critical for the operation of some piece of equipment. In this paper we present a polynomial time algorithm for optimally scheduling instructions with or without time critical con ...

16 Experience with a software-defined machine architecture

David W. Wall

May 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 14 Issue 3

Full text available: pdf(2.86 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We have built a system in which the compiler back end and the linker work together to present an abstract machine at a considerably higher level than the actual machine. The intermediate language translated by the back end is the target language of all high-level compilers and is also the only assembly language generally available. This lets us do intermodule register allocation, which would be harder if some of the code in the program had come from a traditional assembler, out of sight of ...

Keywords: RISC, graph coloring, intermediate language, interprocedural, optimization, pipeline scheduling, profiling, register allocation, register windows

17 Session 10B: VLIW exploration and deisgn synthesis: Power exploration for embedded VLIW architectures

Mariagiovanna Sami, Donatella Sciuto, Cristina Silvano, Vittorio Zaccaria

November 2000 **Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design**

Full text available: pdf(280.04 KB) Additional Information: full citation, abstract, references, citings

In this paper, we propose a system-level power exploration methodology for embedded VLIW architectures based on an instruction-level analysis. The instruction-level energy model targets a general pipeline scalar processor; several architectural parameters such as number and type of pipeline stages as well as average stall/latency cycles per instruction and inter-instruction effects are taken into account. The application of the proposed model to VLIW processors results intractable from the point ...

18 Efficient instruction scheduling using finite state automata

Vasanth Bala, Norman Rubin

December 1995 **Proceedings of the 28th annual international symposium on Microarchitecture**

Full text available: pdf(1.34 MB) Additional Information: full citation, references, citings, index terms

19 Static resource models for code-size efficient embedded processors

Qin Zhao, Bart Mesman, Twan Basten

May 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 2

Full text available: pdf(651.62 KB) Additional Information: full citation, abstract, references, index terms

Due to an increasing need for flexibility, embedded systems embody more and more programmable processors as their core components. Due to silicon area and power considerations, the corresponding instruction sets are often highly encoded to minimize code size for given performance requirements. This has hampered the development of robust optimizing compilers because the resulting irregular instruction set architectures are far from convenient compiler targets. Among other considerations, they int ...

Keywords: Static resource models, constraint analysis, convex hull, phase coupling, scheduling

20 Available instruction-level parallelism for superscalar and superpipelined machines

N. P. Jouppi, D. W. Wall

April 1989 **ACM SIGARCH Computer Architecture News , Proceedings of the third international conference on Architectural support for programming languages and operating systems**, Volume 17 Issue 2

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms

Superscalar machines can issue several instructions per cycle. Superpipelined machines can issue only one instruction per cycle, but they have cycle times shorter than the latency of any functional unit. In this paper these two techniques are shown to be roughly equivalent ways of exploiting instruction-level parallelism. A parameterizable code reorganization and simulation system was developed and used to measure instruction-level parallelism for a series of benchmarks. Results of these si ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next